(19) JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: **10083308 A**

(43) Date of publication of application: **31 . 03 . 98**

(51) Int. Cl **G06F 9/44**
G06F 9/445
G06F 9/46

(21) Application number: **09118665**

(22) Date of filing: **23 . 04 . 97**

(30) Priority: **23 . 04 . 96 US 96 636706**

(71) Applicant: **SUN MICROSYST INC**

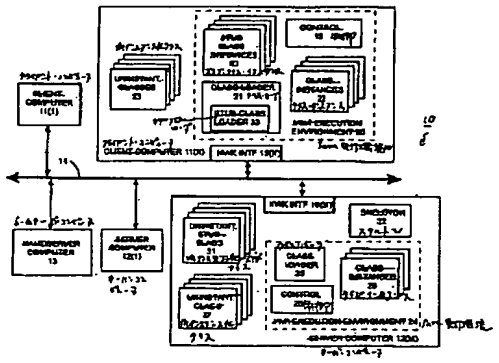
(72) Inventor: **WOLLRATH ANN M**
WALDO JAMES H
RIGGS ROGER

(54) **SUBSYSTEM, METHOD, AND RECORDING MEDIUM FOR STAB RETRIEVAL AND LOADING**

(57) Abstract:

PROBLEM TO BE SOLVED: To make it possible to use a stub together with a remote method call system by providing a stub retriever which starts inspecting the stub and a stub loader which makes the stub usable to remotely call a remote method.

SOLUTION: Each stub class instance 30 is an instance of an uninstanced stub class 31 and made usable by a client computer 11(n) to remotely call a method that a server computer 12(m) itself gives. For various class instances 26 and uninstanced classes 27, this stub class instance is maintained. Then the uninstanced stub class 31 includes a declaration regarding an instance implementing a remote method to be called, and give a method for facilitating access to the remote method or calls it.



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平10－83308

(43)公開日　平成10年(1998)３月31日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ	9/44	５３０		Ｇ０６Ｆ	9/44	５３０Ｍ
	9/445				9/46	３４０Ａ
	9/46	３４０			9/06	４２０Ｃ

審査請求　未請求　請求項の数15　ＦＤ　（全 22 頁）

(21)出願番号　特願平9－118665

(22)出願日　平成９年(1997)４月23日

(31)優先権主張番号　０８／６３６７０６

(32)優先日　1996年４月23日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　591064003
サン・マイクロシステムズ・インコーポレーテッド
SUN MICROSYSTEMS, INCORPORATED
アメリカ合衆国　94303　カリフォルニア州・パロ　アルト・サン　アントニオ　ロード・901

(72)発明者　アン・エム・ウルレース
アメリカ合衆国・01450・マサチューセッツ州・グロートン・ノースウッズ・ロード・9

(74)代理人　弁理士　山川　政樹

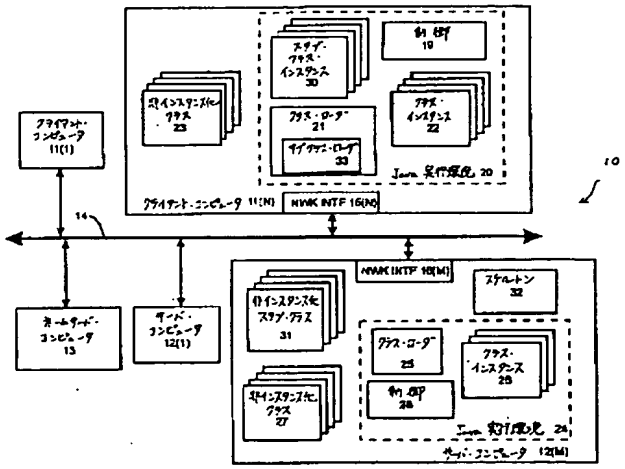
最終頁に続く

(54)【発明の名称】　スタブ検索及びローディング・サブシステム、スタブ検索及びローディング方法並びにスタブ検索及びローディング用記録媒体

(57)【要約】

【課題】　リモート・メソッド呼出しシステムと共に使用されるスタブ検索ローディング・サブシステムを提案する。

【解決手段】　スタブ検索サブシステムが、スタブの検索を開始するスタブ・リトリーバと、スタブがスタブ・リトリーバによって受信されたときに、スタブを実行環境にロードし、それによってスタブをリモート・メソッドのリモート呼出しで使用できるようにするスタブ・ローダとを含む。スタブ検索ローディング・サブシステムは、あるコンピュータから与えられたあるアドレス空間で動作するプログラムのためのスタブ・クラス・インスタンスの検索およびローディングを行い、同じコンピュータから与えることも、あるいは異なるコンピュータから与えることもできる他のアドレス空間で動作するオブジェクトから与えられるメソッドのリモート呼出しを行う。



1

【特許請求の範囲】

【請求項１】　リモート・メソッド呼出しシステムと共に使用され、リモート・メソッドのスタブの検索及び実行環境へのローディングを、前記実行環境で実行されているプログラムによるリモート・メソッドの呼出しを容易にするように制御するスタブ検索及びローディング・サブシステムであって、

Ａ．前記スタブの検索を開始するスタブ・リトリーバと、

Ｂ．前記スタブが前記スタブ・リトリーバによって受信されたときに、前記スタブを前記実行環境にロードし、それによってスタブを前記リモート・メソッドのリモート呼出しで使用できるようにするスタブ・ローダとを備えることを特徴とするスタブ検索及びローディング・サブシステム。

【請求項２】　さらに、リモート・メソッド参照が前記実行環境で受信されたかどうかを検出するリモート・メソッド参照検出器を含み、スタブ・リトリーバが、リモート・メソッド参照が前記実行環境で受信されたことをリモート・メソッド参照検出器が検出したときに前記スタブの検索を開始することを特徴とする請求項１に記載のスタブ検索及びローディング・サブシステム。

【請求項３】　さらに、前記リモート・メソッドの呼出しを制御するリモート・メソッド呼出し制御機構を含み、前記スタブ・リトリーバが、リモート・メソッドが呼び出されたときに前記スタブの検索を開始することを特徴とする請求項１に記載のスタブ検索及びローディング・サブシステム。

【請求項４】　リモート・メソッド呼出しシステムが、さらに、前記リモート・メソッドの処理要求に応答して前記リモート・メソッドを処理し、さらに前記スタブ・リトリーバからの検索要求に応答して前記スタブを与えるサーバを含むことを特徴とする請求項１に記載のスタブ検索及びローディング・サブシステム。

【請求項５】　リモート・メソッド呼出しメソッドと共に使用され、リモート・メソッドのスタブの検索および実行環境へのローディングを容易にし、前記実行環境で実行されているプログラムによるリモート・メソッドの呼出しを容易にするスタブ検索及びローディング方法であって、

Ａ．前記スタブの検索を開始するスタブ検索ステップと、

Ｂ．前記スタブが受信されたときに、前記スタブを前記実行環境にロードし、それによってスタブを前記リモート・メソッドのリモート呼出しで使用できるようにするスタブ・ローディング・ステップとを含むことを特徴とするスタブ検索及びローディング方法。

【請求項６】　さらに、リモート・メソッド参照が前記実行環境で受信されたかどうかを検出するリモート・メソッド参照検出ステップを含み、スタブ検索ステップが、リモート・メソッド参照が前記実行環境で受信されたときに前記スタブの検索を開始するステップを含むことを特徴とする請求項５に記載のスタブ検索及びローディング方法。

2

【請求項７】　さらに、前記リモート・メソッドの呼出しを制御するリモート・メソッド呼出し制御ステップを含み、前記スタブ検索ステップが、リモート・メソッドが呼び出されたときに前記スタブの検索を開始するステップを含むことを特徴とする請求項５に記載のスタブ検索及びローディング方法。

【請求項８】　リモート・メソッド呼出しシステムが、さらに、前記リモート・メソッドの処理要求に応答して前記リモート・メソッドを処理し、さらに前記スタブ・リトリーバからの検索要求に応答して前記スタブを与えるサーバを含むことを特徴とする請求項５に記載のスタブ検索及びローディング方法。

【請求項９】　リモート・メソッド呼出しシステムと共に使用され、リモート・メソッドのスタブの検索および実行環境へのローディングを、前記実行環境で実行されているプログラムによるリモート・メソッドの呼出しを容易にするように制御するようコンピュータを制御するスタブ検索及びローディング用コンピュータで読取り可能な記録媒体であって、

Ａ．前記コンピュータが前記スタブの検索を開始できるようにするスタブ・リトリーバ・コード・デバイスと、

Ｂ．前記スタブが受信されたときに、前記コンピュータが前記スタブを前記実行環境にロードし、それによってスタブを前記リモート・メソッドのリモート呼出しで使用できるようにするスタブ・ローダ・コード・デバイスとがプログラムされたコンピュータ読取り可能な媒体。

【請求項１０】　さらに、前記コンピュータが、リモート・メソッド参照が前記実行環境で受信されたかどうかを検出できるようにするリモート・メソッド参照検出器コード・デバイスを含み、スタブ・リトリーバ・コード・デバイスが、リモート・メソッド参照検出器コード・デバイスによって、前記コンピュータが、リモート・メソッド参照が前記実行環境で受信されたことを検出できるようになったときに前記スタブの検索を開始できるようにすることを特徴とする請求項９に記載のコンピュータ読取り可能な媒体。

【請求項１１】　さらに、前記コンピュータが前記リモート・メソッドの呼出しを制御できるようにするリモート・メソッド呼出し制御コード・デバイスを含み、前記スタブ・リトリーバ・コード・デバイスによって、前記コンピュータが、リモート・メソッドが呼び出されたときに前記スタブの検索を開始することができるようにすることを特徴とする請求項９に記載のコンピュータ読取り可能な媒体。

【請求項１２】　リモート・メソッド呼出しシステムがさらに、前記リモート・メソッドの処理要求に応答して

前記リモート・メソッドを処理し、さらに前記スタブ・リトリーバからの検索要求に応答して前記スタブを与えるサーバを含むことを特徴とする請求項9に記載のコンピュータ読取り可能な媒体。

【請求項13】　リモート・メソッド呼出しシステムと共に使用され、リモート・メソッドのスタブの検索および実行環境へのローディングを、前記実行環境で実行されているプログラムによるリモート・メソッドの呼出しを容易にするように制御するスタブ検索及びローディング・サブシステムであって、

A．コンピュータと、

B．前記コンピュータを制御し、

i．前記スタブの検索を開始するように前記コンピュータを制御するスタブ検索モジュールと、

ii．前記スタブ検索モジュールに応答して前記スタブが受信されたときに、前記スタブを前記実行環境にロードし、それによってスタブを前記リモート・メソッドのリモート呼出しで使用できるようにするよう前記コンピュータを制御するスタブ・ローダ・モジュールとを備える制御構成とを備えることを特徴とするスタブ検索サブシステム。

【請求項14】　リモート・メソッドのスタブの検索および実行環境へのローディングを、前記実行環境で実行されているプログラムによるリモート・メソッドの呼出しを容易にするために制御するコンピュータと共に使用される制御構成であって、

i．前記スタブの検索を開始するように前記コンピュータを制御するスタブ検索モジュールと、

ii．前記スタブ検索モジュールに応答して前記スタブが受信されたときに、前記スタブを前記実行環境にロードし、それによってスタブを前記リモート・メソッドのリモート呼出しで使用できるようにするよう前記コンピュータを制御するスタブ・ローダ・モジュールとを備えることを特徴とする制御構成。

【請求項15】　コンピュータ読取り可能媒体上に記憶され、コンピュータによって実行することができ、それぞれ、リモート・メソッドのスタブの検索および実行環境へのローディングを容易にして前記実行環境で実行されているプログラムによるリモート・メソッドの呼出しを容易にするようコンピュータを制御するように構成された、複数のモジュールを含む、コードを分配するシステムであって、

i．前記スタブの検索を開始するように前記コンピュータを制御するスタブ検索モジュールと、

ii．前記スタブ検索モジュールに応答して前記スタブが受信されたときに、前記スタブを前記実行環境にロードし、それによってスタブを前記リモート・メソッドのリモート呼出しで使用できるようにするよう前記コンピュータを制御するスタブ・ローダ・モジュールとを備えることを特徴とするシステム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、全般的にはディジタル・コンピュータ・システムの分野に関し、詳細には、同じコンピュータ上で実施することも、あるいは他のコンピュータ上でも実施することもできる、コンピュータによってあるアドレス空間内で処理されているプログラムによる、他のアドレス空間でのメソッドおよび手続きの処理の呼出しを容易にするシステムおよび方法に関する。

【0002】本発明は特に、あるアドレス空間で動作するプログラムによる、場合によっては他のコンピュータ上の他のアドレス空間内のリモート・メソッドまたは手続きの呼出しを容易にする「スタブ」情報を得て動的にロードするシステムおよび方法を提供する。

引用による組み込み

The Java™Language Specification（SunMicrosystems, Inc.、1993年ないし1995年）（下記では「Java言語仕様」と呼ぶ）。The Java Virtual Machine Specification（Sun Microsystems, Inc.、1993年ないし1995年）（下記では「Java仮想マシン仕様」と呼ぶ）。Ann Wollrath等著「A Distributed ObjectModel for Java™」。「System And Method For Generating Identifiers For Uniquely IdentifyingObject Types For Objects Used In Processing Of Object−Oriented ProgramsAnd The Like」（Atty. Docket No. P1091）と題する、James H. Waldo、Krishna Bharat、Roger Riggsの名義で本明細書と同日に出願された米国特許出願第08／636，707号。

【0003】

【従来の技術】現代の「企業内」コンピューティングでは、いくつかのパーソナル・コンピュータと、ワークステーションと、その他の装置が、通常、1つまたは複数のコンピュータ・ネットワークとして相互接続される。その他の装置としては、大容量記憶サブシステム、ネットワーク・プリンタ、公衆電話システムとのインタフェースなどがある。パーソナル・コンピュータおよびワークステーションは、ネットワーク大容量記憶サブシステムに記憶できるデータおよびプログラムに関連する処理を実行するために個別のユーザによって使用される。そのような構成では、パーソナル・コンピュータ／ワークステーションは、クライアントとして動作し、通常、データおよびプログラムを処理するためにネットワーク大

5

容量記憶サブシステムからダウンロードする。また、パーソナル・コンピュータまたはワークステーションによって、処理済みのデータを、記憶のためにネットワーク大容量記憶サブシステムにアップロードし、印刷のためにネットワーク・プリンタにアップロードし、公衆電話システムを介した伝送のために電話インタフェースにアップロードすることなどができる。そのような構成では、ネットワーク大容量記憶サブシステム、ネットワーク・プリンタ、電話インタフェースは、ネットワーク内のすべてのクライアントからの要求を実行するために使用できるので、サーバとして働く。ネットワークをそのように構成することによって、サーバはネットワーク内のすべてのパーソナル・コンピュータ／ワークステーションによって容易に使用することができる。そのようなネットワークは、電線や光ファイバなどの通信リンクによってパーソナル・コンピュータ／ワークステーションを相互接続することにより、かなり広い領域にわたって広げることができる。

【０００４】クライアントは、情報を処理のためにサーバからダウンロードするだけでなく、プログラムを処理しながら、クライアントから与えられるある種の「パラメータ」情報に関連するサーバ・コンピュータによる特定のルーチンおよび手続き（一般に「手続き」）の処理をリモートに開始することができる。サーバは、この手続きを処理した後、処理結果をクライアントに与え、クライアントはその後、サーバの処理動作を使用することができる。通常、そのような「リモート手続き呼出し」では、プログラムは、ローカル「スタブ」を使用する。ローカル・スタブは、呼び出されると、特定の手続きを実施し、結果を受信しプログラムに与えるサーバへ要求を転送する。従来、スタブはプログラムと共にコンパイルしなければならず、その場合、リモート手続きを呼び出すのに必要な情報は、プログラムの実行時ではなくコンパイル時に決定しなければならない。クライアントのプログラムが使用できるスタブは、静的なものなので、プログラムのコンパイル時にプログラムに与えるべきスタブに最も近いと判定できるものに過ぎない。したがって、プログラムに与えられるスタブと、プログラムの実行時に呼び出されるリモート手続きの要件との間の不一致のためにエラーが発生し効率が低下する恐れがある。

【０００５】

【発明が解決しようとする課題】本発明は、あるアドレス空間で動作するプログラムが、他のアドレス空間内のメソッドまたは手続きの処理をリモートで呼び出せるようにするために与えられるスタブを得ることができ、かつ動的にロードできるようにし、それによって、スタブを、プログラムのコンパイル時に静的に決定するのではなく、プログラムが実行され必要とされるときにプログラムによってロードできるようにする、改良された新規のシステムおよび方法を提供する。

6

【０００６】

【課題を解決するための手段】ロードされるスタブは、リモート・メソッドまたは手続きを与えるリソースから得ることができ、したがってスタブはリモート・メソッドまたは手続きの呼出し要件を厳密に定義することができる。スタブは、プログラムのコンパイル時に静的に決定されるのではなく、プログラムの実行中に配置し、動的にロードすることができるので、与えられるスタブと、呼び出されるリモート・メソッドまたは手続きの要件との間の不一致のために生じる実行時エラーおよび効率低下を最小限に抑えることができる。

【０００７】簡単に言えば、本発明は、リモート・メソッド呼出しシステムと共に使用できるスタブ検索及びローディング・サブシステムである。スタブ検索及びローディング・サブシステムは、リモート・メソッドに関するスタブの検索および実行環境へのロードを制御し、実行環境内で実行しているプログラムによるリモート・メソッドの呼出しを容易にする。スタブ検索サブシステムは、スタブの検査を開始するスタブ・リトリーバと、スタブ・リトリーバからスタブを受信したときにスタブを実行環境にロードし、それによってスタブをリモート・メソッドのリモート呼出しに使用できるようにするスタブ・ローダとを含む。一実施態様では、スタブ検索及びローディング・サブシステムは、あるコンピュータから与えられるあるアドレス空間で動作するプログラムに関するスタブ・クラス・インスタンスの検索およびローディングを実行し、同じコンピュータまたは異なるコンピュータから与えることができる他のアドレス空間で動作するオブジェクトから与えられるメソッドのリモート呼出しを実行する。これと同じ実施態様で、スタブ検索及びローディング・サブシステムは、リモート・オブジェクトの参照時にスタブ・クラス・インスタンスの検索およびローディングを実行する。ただし、他の実施態様では、リモート・メソッドの呼出し時に、検索およびローディングを実行することができる。

【０００８】

【発明の実施の形態】本発明の上記およびその他の利点は、下記の説明を添付の図面と共に参照することによってよりよく理解することができる。図１は、「スタブ」情報の動的ローディングを容易にし、あるアドレス空間で動作するプログラムが他のアドレス空間内のメソッドまたは手続きの処理をリモート的に呼び出すことができるようにする構成を含むコンピューク・ネットワーク１０の概略図である。図１を参照すると分かるように、コンピュータ・ネットワーク１０は、複数のクライアント・コンピュータ１１（１）ないし１１（Ｎ）（一般に参照符号１１（ｎ）によって識別される）と、複数のサーバ・コンピュータ１２（１）ないし１２（Ｍ）（一般に参照符号１２（ｍ）によって識別される）とを含み、これらのコンピュータはすべて、通信リンク１４で表され

たネットワークによって相互接続される。また、ネットワーク１０は、やはり通信リンク１４に接続することができる少なくとも１つのネームサーバ・コンピュータ１３を含むことができる。ネープサーバ・コンピュータ１３の目的については下記で説明する。従来どおり、少なくともいくつかのクライアント・コンピュータ１１（ｎ）は、それぞれ、通常システム装置と、ビデオ表示装置と、キーボードやマウスなどのオペレータ入力装置（すべてが独立に示されているわけではない）とを含む、パーソナル・コンピュータまたはコンピュータ・ワークステーションの形である。サーバ・コンピュータ１２（ｍ）およびネームサーバ・コンピュータ１３は通常、システム装置（やはり独立には示していない）も含み、ビデオ表示装置とオペレータ入力装置を含むこともできる。

【０００９】クライアント・コンピュータ１１（ｎ）、サーバ・コンピュータ１２（ｍ）、ネームサーバ・コンピュータ１３はすべて、従来型の記憶済みプログラム・コンピュータ・アーキテクチャのものである。システム装置は一般に、ディスク記憶要素やテープ記憶要素などの処理装置、メモリ装置、大容量記憶装置と、それぞれのコンピュータと通信リンク１４を相互接続するネットワーク・インタフェース装置１５（ｎ）、１６（ｍ）を含め他の要素（独立には示していない）とを含む。ビデオ表示装置によって、コンピュータは処理済みデータおよび処理状況をオペレータに表示することができ、オペレータ入力装置によって、オペレータはコンピュータによるデータ処理および制御処理を入力することができる。コンピュータ１１（ｎ）および１２（ｍ）および１３は、それぞれのネットワーク・インタフェース装置１５（ｎ）、１６（ｍ）を通じ、通信リンク１４を介して、情報をメッセージの形で互いに転送し合う。

【００１０】一実施形態では、ネットワーク１０は「クライアントーサーバ」構成として構成され、図１にコンピュータ１２（ｍ）として示された１つまたは複数のコンピュータがサーバとして動作し、図１にコンピュータ１１（ｎ）として示した他のコンピュータがクライアントとして動作する。一態様では、１つまたは複数のサーバ・コンピュータ１２（ｍ）は、「ファイル・サーバ」として大容量記憶装置を含むことができる。大容量記憶装置は、クライアント・コンピュータ自体の処理動作で使用できるように通信リンク１３を介してクライアント・コンピュークによって検索できるプログラムおよびデータのコピーを記憶することができる。クライアント・コンピュータ１１（ｎ）は必要に応じて、サーバ・コンピュータ１２上にデータを格納することもできる。このデータは、後で、データを格納したクライアント・コンピュータまたは他のクライアント・コンピュータが、その処理動作で使用できるように検索することもできる。また、１つまたは複数のサーバ・コンピュータ１２

（ｍ）は「コンピュータ・サーバ」として、クライアント・コンピュータ１１（ｎ）からのリモート要求に応答してある種の処理動作を実行し、処理結果を、要求側クライアント・コンピュータ１１（ｎ）が後に続く処理において使用できるようにコンピュータ１１（ｎ）に返すことができる。いずれのケースでも、サーバ・コンピュータは、クライアント・コンピュータ１１（ｎ）にほぼ類似するものであってよく、システム装置と、ビデオ表示装置と、オペレータ入力装置とを含み、オペレータによってクライアント・コンピュータと同様にデータ処理動作に使用することができる。別法として、少なくともいくつかのサーバ・コンピュータは、クライアント・コンピュータからの検索要求または記憶要求またはリモート処理要求を検索して処理し、そのような要求に対応する応答を生成する処理要素、メモリ要素、大容量記憶要素、ネットワーク・インタフェース要素のみを含むことができる。当然のことながら、クライアント・コンピュータ１１（ｎ）は、サーバ・コンピュータ１２（ｍ）によって実行される動作として本明細書に記載された動作を実行することもでき、同様にサーバ・コンピュータ１２（ｍ）は、クライアント・コンピュータ１１（ｎ）によって実行される動作として本明細書に記載された動作を実行することもできる。

【００１１】通信リンク１４で表されたネットワークは、たとえば、通常、個別の企業内に維持されるローカル・エリア・ネットワーク（ＬＡＮ）およびワイド・エリア・ネットワーク（ＷＡＮ）や、公衆電話網や、インターネットや、様々なコンピュータ間でディジタル・データを転送することができる他のネットワークを含め、クライアント・コンピュータ１１（ｎ）、サーバ・コンピュータ１２（ｍ）、ネームサーバ・コンピュータ１３が通信するために使用できるいくつかのタイプのネットワークを備えることができる。ネットワークは、たとえばワイヤや、光ファイバや、無線リンクや、図１に示した様々なコンピュータ間で情報を表す信号を送るためのその他の媒体を含め、いくつかの通信媒体のうちのどれかを使用して実施することができる。前述のように、各コンピュータは通常、それぞれのコンピュータを通信リンク１４に接続し、コンピュータが通信リンクを介して情報を送受できるようにするネットワーク・インタフェースを含む。

【００１２】本発明は、「スタブ」情報を得ること、及び動的にロードすることを容易にして、あるアドレス空間で動作するプログラムが、呼出し側プログラムとして同じコンピュータ上に配置することも、あるいは異なるコンピュータ上に配置することもできる他のアドレス空間内のリモート・メソッドまたは手続きを呼び出すことができるようにする。Ｊａｖａ仮想マシンから与えられる実行環境に関連して処理される、Ｊａｖａ言語仕様で記述されたＪａｖａ™プログラミング言語で与えられた

プログラムに関連して本発明を説明する。Ｊａｖａ仮想マシンは、Ｊａｖａ仮想マシン仕様で指定される。Ｊａｖａ言語仕様で記述されたＪａｖａプログラミング言語のプログラムは、「クラス」および「インタフェース」を定義する。クラスは１つまたは複数のメソッドまたは手続きを定義するために使用される。メソッド又は手続きはそれぞれ、インタフェースを参照することによって呼び出すことができる。クラスは、「スーパークラス」に関連付けられ、「スーパークラス」を拡張することができ、そのため、スーパークラスのすべてのインタフェースおよびメソッドを備え、追加インタフェースまたはメソッド、あるいはその両方を含むこともできる。クラスは１つまたは複数のサブクラスを有することもできる（したがって、各サブクラスのスーパークラスを備える）。サブクラスはそれぞれ、スーパークラスと一体となり、場合によってはスーパークラスを拡張する。

【００１３】インタフェースは、１組のメソッドを宣言するための機構を与える。なお、インタフェースは、たとえば名前によって、インタフェースによって宣言された各メソッドを識別し、メソッドに与えるべき引数のデータ・タイプ、メソッドから返されるリターン値のデータ・タイプ、メソッドの処理中に破棄できる例外の識別子を識別する。クラスは、特定のインタフェースを実施することを示すことができ、そのため、インタフェースで宣言されるすべてのメソッドの処理で使用されるプログラム・コードを含む。また、それぞれの異なるクラスは、同じインタフェースを実施することを示すことができ、各クラスは、インタフェースで宣言されたすべてのメソッドを処理する際に使用されるプログラム・コードを有するが、メソッドを処理する際に使用できるように各クラスで与えられるプログラム・コードは、同じメソッドを処理する際に使用される、他のクラスで与えられるプログラム・コードとは異なることがある。したがって、インクフェースは、メソッドの処理で使用される手続きを示すものを与えずに１組のメソッドを宣言するための機構を与える。インタフェースは、そのインタフェースを使用して呼び出すことができるメソッドを実施する特定のクラスとは独立に宣言することができる。なお、メソッドを呼び出すクラスと、メソッドを実際に実施するクラスは、共通のスーパークラスを共用する必要はない。

【００１４】Ｊａｖａ仮想マシン仕様で記述されたＪａｖａプログラムの処理時に、クライアント・コンピュータ１１（ｎ）は、Ｊａｖａプログラムを解釈するための実行環境２０を与える。Ｊａｖａ仮想マシンは、プログラムの実行中に、制御モジュール１９の制御下で、図１で全体的に参照符号２２で識別されたクラスのインスタンスを動的に、動作中のプログラムの実行環境にリンクすることができる。その動作で、制御モジュール１９は実際上、Ｊａｖａプログラムの実行時に、全体的に参照符号２３で識別されたそれぞれの非インスタンス化クラスが実施するメソッドが呼び出されるとき、クラス・ローダがクラス２３を検索し、それをインスタンス化し、クラス・インスタンス２２として実行環境のアドレス空間にリンクすることができるようにする。また、クラス・ローダ２１は、クラス・インスタンス２２が必要でないときに、あるいはメモリを節約するために、クラス・インスタンス２２を破棄することができる。当然のことながら、クラス・インスタンス２２が破棄された場合、後で必要になった場合にはクラス・ローダ２１によって再ロードすることができる。

【００１５】本発明は、クライアント・コンピュータ１１（ｎ）によって実行環境２０で実行しているプログラムによる、サーバ・コンピュータ１２（ｍ）上のクラスによって実施されるメソッドのリモート呼出しを容易にする構成を提供する。サーバ・コンピュータ１２（ｍ）は、メソッドを実行する際、制御モジュール２８の制御下でＪａｖａメソッドを処理するための実行環境２４も与える。その動作で、実行環境２１を与えるＪａｖａ仮想マシンは、制御モジュール２８の制御下で、クラス２６のインスタンスと、リモートに呼び出されるメソッドを処理するために必要な他のクラスのインスタンス（やはり全体として参照符号２６で表される）を動的にリンクし、メソッドを実行環境２４で処理できるようにすることができるクラス・ローダ２５（クラス・ローダ２１と同様なものでよい）を含む。その動作で、制御モジュール２８は実際上、クラス・ローダ２５が、全体として参照符号２７で識別された非インスタンス化クラスから、呼び出すべきメソッドの非インスタンス化クラスを検索し、それ（すなわち、呼び出すべきメソッドを提供する非インスタンス化クラス）をインスタンス化し、クラス・インスタンス２６として実行環境にリンクできるようにする。また、クラス・ローダ２５は、メソッドの処理が終了したときにクラス・インスタンス２６を破棄することができる。当然のことながら、クラス・インスタンス２６が破棄された場合、後で必要になった場合にはクラス・ローダ２５によって再ロードすることができる。

【００１６】ネームサーバ・コンピュータ１３が設けられている場合、その構造は一般に、サーバ・コンピュータ１２（ｍ）の構造に類似しており、独立には説明しない。

【００１７】クライアント・コンピュータの実行環境２１の制御モジュール１９は、メソッドのリモート呼出しを容易にするために、リモート・メソッドを呼び出しているクラス・インスタンスを含め様々なクラス・インスタンス２２が処理される実行環境２１の一部として設けられた、全体として参照符号３０で識別された１つまたは複数のスタブ・クラス・インスタンスを使用する。各スタブ・クラス・インスタンス３０は、非インスタンス

化スタブ・クラス３１のインスタンスであり、サーバ・コンピュータ１２（m）は、それ自体が、「エクスポート」し、すなわちサーバ・コンピュータ１２（m）自体が与えるメソッドのリモート呼び出しにおいてクライアント・コンピュータ１１（n）によって使用できるようにする、様々なクラス・インスタンス２６および非インスタンス化クラス２７のために、このスタブ・クラス・インスタンスを維持することができる。非インスタンス化スタブ・クラス３１は、呼び出すべきリモート・メソッドを実施する特定のリモート非インスタンス化クラス２７の完全な１組のインタフェースに関する宣言を含み、また、リモート・クラスによって実施されるリモート・メソッドのアクセスを容易にするメソッドを与え、あるいは呼び出す。非インスタンス化スタブ・クラス３１は、インスタンス化されクライアント・コンピュータ１１（n）の実行環境２０にスタブ・クラス・インスタンス３０として与えられたときに、実際上、呼出し側Ｊａｖａプログラムの実行環境２０の制御モジュール１９が必要とする情報を与え、そのため、関連するクラスによって実施されるリモート・メソッドが、特定の実行環境で動作しているＪａｖａプログラムによって呼び出されたときに、リモート・メソッドが処理され、呼出し側Ｊａｖａプログラムにリターン値が与えられる。一実施形態では、スタブ・クラス・インスタンスを実行環境２０に与えるための構成は、前述のＷａｌｄｏ等の特許出願に記載された構成に類似している。

【００１８】また、サーバ・コンピュータ１２（m）は、サーバ・コンピュータ１２（m）からエクスポートされた特定のクラスおよびメソッドを識別するスケルトン３２と、サーバ・コンピュータ１２（m）がどのように、それぞれのクラスをロードし、サーバ・コンピュータから与えられた特定のメソッドの処理を開始するかに関する情報を与える。

【００１９】クラス・インスタンスは、サーバ・コンピュータ１２（m）によって維持されるリモート・メソッドを呼び出す際、リモート・メソッドが処理において使用する様々なパラメータの値をリモート・メソッドのスタブ・クラス・インスタンス３０に与える。リモート・メソッドが呼出し側Ｊａｖａプログラムと同じコンピュータ上で実施される場合、呼出し側Ｊａｖａプログラムがリモート・メソッドを呼び出す際、コンピュータは、実行環境２０と同様な実行環境を確立し、実行環境のクラス・ローダが、メソッドをクラス・インスタンス２２と同様なクラス・インスタンスとして実施するクラスをロードしインスタンス化することができるようにし、リモート呼出し時に呼出し側クラス・インスタンスから与えられるパラメータの値を使用してリモート・メソッドを処理することができる。メソッドの処理が完了した後、リモート・メソッドが処理された実行環境は、呼出しを行ったリモート・メソッドのスタブ・クラス・インスタンス３０に結果を与え、スタブ・クラス・インスタンス３０は、リモート・メソッドを呼び出した特定のクラス・インスタンス２２に結果を与える。

【００２０】クライアント・コンピュータ１１（n）とサーバ・コンピュータ１２（m）がそれぞれの異なる物理コンピュータ上で実施される場合でも、同様な動作が実行される。その場合、リモート呼出しに応答して、呼出し側クラス・インスタンス２２の実行環境１０の制御モジュール１９の制御下で、呼出し側クラス・インスタンス２２を処理しているクライアント・コンピュータ１１（n）は、適当なスタブ・クラス・インスタンス３０を使用して、通信リンク１４で表されたネットワークを介して、リモート・メソッドを実施するサーバ・コンピュータ１２（m）と通信し、サーバ・コンピュータ（１２）mが、リモート・メソッドを実施するクラスの実行環境２４を確立し、クラス・ローダ２５を使用してこのクラスのインスタンスをクラス・インスタンス２６としてロードすることができるようにする。また、クライアント・コンピュータ１１（n）は、やはり適当なスタブ・クラス・インスタンス３０を使用して、ネットワーク１４を介してサーバ・コンピュータ１２（m）に必要なパラメータ値を与える。その後、サーバ・コンピュータ１２（m）は、そのように与えられたパラメータ値を使用してリモート・メソッドを処理し、ネットワークを介してクライアント・コンピュータ１１（n）、特に適当なスタブ・クラス・インスタンス３０へ転送される結果値を生成する。クライアント・コンピュータ１１（n）は、ネットワークから結果値を受信した後、その結果値を呼出し側クラス・インスタンス２２の処理のためにこのクラス・インスタンス２２に与える。

【００２１】いずれの場合も、クライアント・コンピュータの実行環境２０の制御モジュール１９は、リモート・オブジェクトの参照が受信されたと判定したときに、スタブ・クラス・インスタンス３０が存在していないと判定した場合、たとえばリモート・メソッドを実施するサーバ・コンピュータ１２（m）からスタブ・クラス・インスタンス３０を得て、スタブ・クラス・インスタンス３０を動的に呼出し側クラス・インスタンス２２の実行環境２０にロードできるようにする。リモート・オブジェクトの参照は、たとえば他のリモート・メソッド呼出しのリターン値と他のリモート・メソッド呼出し中に受信されるパラメータのどちらかとして受信することができる。スタブ・クラス・インスタンスは、クラス・インスタンス２２を実行環境２０にロードするために使用される方式と同様な方式で実行環境に動的にロードすることができる。実行環境２０は、制御モジュール１９の制御下で、実行環境で処理されるクラス・インスタンス２２が必要とするスタブ・クラス・インスタンス３０を見つけてロードしようとするスタブ・クラス・ローダ３３を備える。リモート的に呼び出すべきメソッドを実施

13

するクラスを維持する特定のサーバ・コンピュータ１２（m）の位置を、呼出し側クラス・インスタンスからの呼出しに含めることも、あるいはクライアント・コンピュータ１１（n）によって維持される他の機構（図示せず）を通じてスタブ・クラス・ローダ３３に知らせることもできる。

【００２２】しかし、スタブ・クラス・ローダ３３は、リモート的に呼び出すことができるメソッドを実施するクラスをどのサーバ・コンピュータ１２（m）が維持しているかを通知されない場合、ネームサーバ・コンピュータ１３を使用してそのＩＤを与えることができる。このＩＤは、サーバ・コンピュータ１２（m）、またはネットワーク１４上で使用可能であり、あるいはサーバ・コンピュータ１２（m）が応答することができる他のリソースを識別するために使用できる識別子を備えることができる。例示的な識別子はたとえば、サーバ・コンピュータまたはリソース、あるいはその両方を識別するネットワーク・アドレスを含み、あるいはネットワーク１４がインターネットであり、あるいはインターネットを含む場合は、たとえば、ＩＤを与えることができるＷｏｒｌｄ　Ｗｉｄｅ　Ｗｅｂリソースの識別子、またはインターネット上で使用できるリソースを識別する一様な機構を与えるＵＲＬを含む。リモート・メソッドを実施するサーバ・コンピュータ１２（m）は、クライアント・コンピュータ１１（n）からの要求に応答してスタブ・クラス・インスタンス３０を与え、クライアント・コンピュータ１１（n）はこのスタブ・クラス・インスタンス３０を実行環境２１にロードし、その後リモート呼出しを開始できるようにすることができる。

【００２３】前述のように、スタブ・クラス・ローダ３３は、どのサーバ・コンピュータ１２（m）が、呼び出すことができるリモート・メソッドを実施するのか分からない（したがって、どのコンピュークがリモート呼出し用のスタブ・クラス・コードを与えるのか分からない）場合、制御モジュール１９の制御下で、ネームサーバ・コンピュータ１３からＩＤを得る。その動作で、スタブ・クラス・ローダ３３は、そのようなケースで使用するために与えられる事前に与えられたデフォルト・スタブ・クラスを使用することができる。デフォルト・クラス・スタブは、呼出し側Ｊａｖａプログラムによって使用されると、呼出し側Ｊａｖａプログラムを処理するコンピュータが、ネームサーバ・コンピュータ１３と通信し、リモート・メソッドを呼び出す際に使用できる情報を得ることができるようにする。この動作は、ネームサーバ・コンピュータ１３によって処理すべきリモート・メソッドの呼出しとほぼ同じであり、リモート・メソッドは、リモート的に呼び出すべきクラスおよびメソッドを識別し、ネームサーバ・コンピュータ１３が、要求側クライアント・コンピュータ１１（n）に、メソッドを処理することができるサーバ・コンピュータ１２

14

（m）のＩＤを与え、かつサーバ・コンピュータ１２（m）と通信し、特定のメソッドを呼び出すうえで助けとなるその他の情報を与えることができるようにするパラメータを含む。当然のことながら、ネームサーバ・コンピュータ１３は、「エクスポートされたリソース」、すなわちネットワーク１４に接続されたクライアント・コンピュータ１１（n）が使用できるクラスやメソッドなどのリソースと、エクスポートされたリソースを使用する際にクライアント・コンピュータ１１（n）に有用な、そのようなリソースを与える特定のサーバ・コンピュータ１２（m）のＩＤなどの情報とのテーブル（独立には示していない）を維持する。

【００２４】当然のことながら、ネームサーバ・コンピュータ１３は、当技術分野で知られているいくつかの方法でエクスポート・リソース・テーブルを作成し維持することができる。たとえば、ネームサーバ・コンピュータ１３は、ネットワーク１４を介して、エクスポート・リソース情報を求める要求を定期的に同報通信することができ、この要求に対して、エクスポートされたリソースを維持する様々なサーバ・コンピュータ1.2（m）が応答することができる。その場合、ネームサーバ・コンピュータ１３は、サーバ・コンピュータ１２（m）からの応答に基づいて、エクスポート・リソース・テーブルを確立することができる。別法として、エクスポートされたリソースを維持する様々なサーバ・コンピュータ１２（m）はそれぞれ、それが維持するエクスポートされたリソースに関する情報を定期的に同報通信することができ、ネームサーバ・コンピュータ１３は、サーバ・コンピュータからの同報通信に基づいてエクスポート・リソース・テーブルを更新することができる。また、ネームサーバ・コンピュータのエクスポート・リソース・テーブルは、システム・オペレータによって確立することができ、システム・オペレータが更新するまで固定することができる。

【００２５】いずれの場合も、デフォルト・スタブによって開始された要求に応答してネームサーバ・コンピュータ１３から与えられる情報には、たとえば、呼び出すべきリモート・メソッドを実施するクラスを与えることができるコンピュータ１２（m）のＩＤなどの情報と、必要なスタブ・クラス・コードなどを与えるためにコンピュータ（すなわち、リモート・メソッドを実施するコンピュータ）によって必要とされる特定の情報とが含まれる。呼出し側Ｊａｖａプログラムを処理しているコンピュータ１１（n）は、ネームサーバ・コンピュータ１３から情報を受信した後、制御モジュール１９の制御下で、この情報を使用して、コンピュータ（すなわち、リモート・メソッドを実施するコンピュータ）と通信しスタブ・クラスを得て、その後前述のようにメソッドを呼び出すことができる。

【００２６】この背景情報を用いて、リモート・メソッ

15

ドへの参照を受信したときにスタブ・クラス・インスタンスを得て動的にロードすることに関連してクライアント・コンピュータ１１（ｎ）と、サーバ・コンピュータ１２（ｍ）と、必要に応じてネームサーバ１３によって実行される動作について、図２－図４に示したフローチャートに関連して説明する。また、スタブ・クラス・インスタンスを使用するメソッドのリモート呼出しに関連してクライアント・コンピュータ１１（ｎ）およびサーバ・コンピュータによって実行される動作について、図５、図６に示したフローチャートに関連して説明する。最初に図２－４を参照すると分かるように、実行環境制御モジュール１９は、リモート・メソッドへの参照を受信すると最初に、リモート・メソッドの呼出しを容易にするために実行環境２０に適当なスタブ・クラス・インスタンスが存在するかどうかを判定する（ステップ１００）。制御モジュール１９は、実行環境にリモート・メソッドのそのようなスタブ・クラス・インスタンス３０が存在すると判定した場合、他の動作を継続する（ステップ１０１）。しかし、制御モジュール１９は、ステップ１０１で、リモート・メソッドの実行環境２０にそのようなスタブ・クラス・インスタンスが存在しないと判定した場合、スタブ・クラス・ローダ３３を使用して、リモート・メソッドを処理するクラスのスタブ・クラス・インスタンス３０を見つけてロードしようとする。その場合、制御モジュール１９は最初、クラス・インスタンス２２からの呼出しに、呼び出すべきメソッドのクラスを維持するサーバ・コンピュータ１２（ｍ）または他のリソースを識別するリソース・ロケータが含まれていたかどうか、あるいは制御モジュール１９またはスタブ・クラス・ローダ３３がそのようなリソース・ロケータを備えているかどうかを判定する（ステップ１０２）。制御モジュール１９は、そのステップで肯定的な判定を下す場合、ステップ１０３へ進み、スタブ・クラス・ローダ３３が、識別されたサーバ・コンピュータ１２（ｍ）との通信を開始し呼び出すべきクラスおよびメソッドのスタブ・クラス・インスタンスを得ることができるようにする（ステップ１０３）。スタブ・クラス・ローダ３３は、サーバ・コンピュータ１２（ｍ）からスタブ・クラス・インスタンス３０を受信すると、ステップ１００でリモート・メソッド呼出しを開始したクラス・インスタンス２１の実行環境２０にスタブ・クラス・インスタンス３０をロードする（ステップ１０４）。参照されたリモート・メソッドのスタブ・クラス・インスタンス３０が実行環境にロードされた後、下記で図５、６に関連して説明するようにメソッドを呼び出すことができる。

【００２７】ステップ１０２に戻ると分かるように、制御モジュール１９が、クラス・インスタンス２２からの呼出しに、呼び出すべきメソッドのクラスを維持するサーバ・コンピュータ１２（ｍ）または他のリソースを識

16

別するリソース・ロケータが含まれておらず、さらに、制御モジュール１９もスタブ・クラス・ローダ３３もそのようなリソース・ロケータを備えていないと判定した場合、「ｃｌａｓｓｎｏｔ　ｆｏｕｎｄ」例外を示すことができ、その点で、制御モジュール１９は、例外ハンドラを呼び出すことができる。例外ハンドラは、たとえば、単にリモート・メソッドが見つからなかったことを制御モジュール１９に知らせることなどを含め、いくつかの回復動作のうちのどれかを実行し、制御モジュールが後に続く動作を決定できるようにする。

【００２８】別法として、制御モジュール１９は、たとえばデフォルト・スタブ・クラス・インスタンス３０の呼出しを使用して、ネットワーク１４から与えられるネームサーバ・コンピュータ１３またはその他のリソース（図１に全体としてネームサーバ・コンピュータ１３で表されている）からリソース・ロケータを得ようとすることができる。デフォルト・スタブ・クラス・インスタンス３０の呼出しは、呼び出すべきクラスおよびメソッドのＩＤと、ネームサーバ・コンピュータ１３（ｍ）の名前を含む。制御モジュール１９は、デフォルト・スタブ・クラス・インスタンス３０を使用して、コンピュータ１１（ｎ）がネームサーバ・コンピュータ１３との通信を開始し、呼び出すべきクラスおよびメソッドを維持するサーバ・コンピュータ１２（ｍ）の識別子を得ることができるようにする（ステップ１１０）。デフォルト・スタブ・クラス・インスタンス３０からの通信は基本的に、リモート・メソッド呼出しに対応する。メソッドによって、ネームサーバ・コンピュータは、リモートに呼び出すべきクラスおよびメソッドに関連付けられたサーバ・コンピュータ１２（ｍ）のＩＤが存在する場合にそれを与え、あるいはクラスおよびメソッドに関連付けられたものとして識別されたサーバ・コンピュータ１２（ｍ）がないことを示す表示を与えることができる。ステップ１１０での通信中、デフォルト・スタブ・クラス・インタフェース３０は、パラメータ値として、呼び出すべきクラスおよびメソッドのＩＤを与える。

【００２９】ネームサーバ・コンピュータ１３は、デフォルト・スタブ・クラス・インスタンス３０からの通信に応答して、要求をリモート・メソッドとして処理する（ステップ１１１）。結果情報は、リモートに呼び出すべきクラスおよびメソッドに関連付けられたサーバ・コンピュータ１２（ｍ）のＩＤが存在する場合にそれを備え、あるいはクラスおよびメソッドに関連付けられたものとして識別されたサーバ・コンピュータ１２（ｍ）がないことを示す表示を備える。ネームサーバ・コンピュータ１３は、メソッドを終了した後、デフォルト・スタブ・クラス・インスタンス３０との通信を開始し、デフォルト・スタブ・クラス・インスタンス３０に結果情報を与える（ステップ１１２）。

【００３０】デフォルト・スタブ・クラス・インスタン

スは、ネームサーバ・コンピュータ１３から結果情報を受信した後、制御モジュール１９の制御下で、スタブ・クラス・ローダ３３に結果情報を渡す（ステップ１１３）。その後、スタブ・クラス・ローダ３３は、ネームサーバ・コンピュータからの結果情報がサーバ・コンピュータ１２（m）のＩＤを備えているか、それともクラスに関連付けられたものとして識別されたサーバ・コンピュータ１２（m）がないことを示す表示を備えているかを判定する（ステップ１１４）。スタブ・クラス・ローダ３３は、結果情報がサーバ・コンピュータ１２（m）のＩＤを備えると判定した場合、ステップ１０１に戻り、識別されたサーバ・コンピュータ１２（m）との通信を開始し、呼び出すことができるクラスおよびメソッドのスタブ・クラス・インスタンスを得る。一方、ネームサーバ・コンピュータ１３が、呼び出すことができるクラスおよびメソッドに関連付けられたものとして識別されたサーバ・コンピュータ１２（m）がないことを示す表示を与えた、とスタブ・クラス・ローダ３３がステップ１１４で判定した場合、前述のように、「ｃｌａｓｓ　ｎｏｔ　ｆｏｕｎｄ」例外を示し（ステップ１１５）、例外ハンドラを呼び出すことができる。

【００３１】前述のように、上記で図２－４に関連して説明したように検索されロードされたスタブ・クラス・インスタンス３０をメソッドのリモート呼出しで使用することができる。メソッドのリモート呼出しに関連してクライアント・コンピュータ１１（n）によって実行される動作を図５，６のフローチャートに関連して説明する。図に示したように、クラス・インスタンス２２がメソッドを呼び出すと、制御モジュール１９は最初、呼び出すべきリモート・メソッドの実行環境にスタブ・クラス・インスタンス３０が存在することを検証する（ステップ１２０）。ステップ１２０で肯定の判定が下された場合、スタブ・クラス・インスタンス３０は、リモート呼出しに使用され、リモート呼出し時に、リモート・メソッドを処理する際に使用されるパラメータ値を与える（ステップ１２１）。その後、呼び出すことができるリモート・メソッドのスタブ・クラス・インスタンス３０を使用して、リモート・メソッドのクラスを維持するサーバ・コンピュータ１２（m）との通信を開始する（ステップ１２２）。このプロセス中に、リモート・メソッドを渡す際に使用される転送パラメータ値が渡される。当然のことながら、メソッドを処理するサーバ・コンピュータ１２（m）が、メソッドを呼び出しているクライアント・コンピュータ１２（n）と同じ物理コンピュータである場合、物理コンピュータ内で処理されている実行環境間で通信を行うことができる。一方、メソッドを処理するサーバ・コンピュータ１２（m）が、メソッドを呼び出しているクライアント・コンピュータ１２（n）の物理コンピュータとは異なる物理コンピュータである場合、通信はクライアント・コンピュータおよびサーバ・コンピュータのそれぞれのネットワーク・インタフェース１５（n）および１６（m）を通じネットワーク１４を介して行われる。

【００３２】サーバ・コンピュータ１２（m）は、ステップ１２２でのスタブ・クラス・インスタンスからの通信に応答し、必要に応じて、呼び出すべきメソッドを維持するクラスの実行環境２４を確立し、スケルトン３２から与えられる情報を使用してそのクラスのクラス・インスタンス２６を作成する（ステップ１２３）。その後、サーバ・コンピュータ１２（m）は、制御モジュール２８の制御下で、スタブ・クラス・インスタンス３０から与えられたパラメータ値に関連してメソッドを処理する（ステップ１２４）。サーバ・コンピュータ１２（m）は、メソッドの処理を完了した後、やはり制御モジュール２８の制御下で、クライアント・コンピュータのスタブ・クラス・インスタンス３０との通信を開始し、スタブ・クラス・インスタンスに結果情報を与える（ステップ１２５）。上記でステップ１０２に関連して説明したのと同様に、メソッドを処理したサーバ・コンピュータ１２（m）が、メソッドを呼び出したクライアント・コンピュータ１２（n）と同じ物理コンピュータである場合、この物理コンピュータ内で処理されている実行環境２４および２０間で通信を行うことができる。メソッドを処理したサーバ・コンピュータ１２（m）が、メソッドを呼び出しているクライアント・コンピュータ１２（n）の物理コンピュータとは異なる物理コンピュータである場合、通信は、サーバ・コンピュータおよびクライアント・コンピュータのそれぞれのネットワーク・インタフェース１６（m）および１５（n）を通じネットワーク１４を介して行われる。スタブ・クラス・インスタンス３０は、サーバ・コンピュータから結果情報を得た後、リモート・メソッド呼出しを開始したクラス・インスタンス２２に結果情報を与えることができ（ステップ１２６）、そのクラス・インスタンス２２は、制御モジュール１９の制御下で処理を継続することができる。

【００３３】ステップ１２０に戻ると分かるように、制御モジュール１９は、そのステップで、呼び出すことができるリモート・メソッドに妥当なスタブ・クラス・インスタンス３０を有さないと判定した場合、その点で例外ハンドラを呼び出し（ステップ１２７）、選択されたエラー回復動作を実行することができる。

【００３４】本発明は、いくつかの利点を与える。特に、本発明は、ある実行環境で動作しているプログラムが、他の実行環境内のメソッドの処理をリモートに呼び出すことができるようにするスタブの動的ローディングを容易にし、それによってプログラムが実行され必要とされるときにスタブをロードできるようにする新しいシステムおよび方法を提供する。スタブがプログラムと共にコンパイルされ、したがってプログラムのコンパイル

時に静的に決定されるシステムでは、スタブは、プログラムが受信するリモート参照によってサポートされる実際の1組のリモート・インタフェースのサブセットを実施することがあり、そのため、プログラムに与えられるスタブと、プログラムの実行時に呼び出されるリモート手続きの要件との間の不一致のためにエラーが発生し効率が低下することがある。しかし、動的スタブ・ローディング・システムおよび方法では、リモート・メソッドを与える特定のリソースから、ロードされるスタブを得ることができるので、スタブは、実行時に呼出し側プログラムに与えるべき厳密な1組のインタフェースを定義し、それによって、与えられるスタブと、呼び出されるリモート・メソッドの要件との間の不一致のために生じる実行時非整合性をなくすことができる。

【００３５】当然のことながら、前述の構成にはいくつかの修正を加えることができる。たとえば、リモート・メソッドへの参照の受信時に、スタブ・クラス・インスタンスを得てロードしリモート・メソッドの呼出しを容易にするものとして実行環境２０を説明したが、当然のことながら、リモート・メソッドが最初に呼び出されたときにスタブ・クラス・インスタンスを得てロードすることができる。リモート。メソッドの参照の受信時にリモート・メソッドのスタブ・クラス・インスタンスを得てロードすることは、（ｉ）リモート・メソッドが実際に呼び出されるときに実行環境にスタブ・クラス・インスタンスが存在し、（ｉｉ）適当なスタブ・クラス・インスタンスをロードできない場合に、プログラムまたはオペレータに早いうちに知らせることができるという利点を有する。一方、メソッドを呼び出すときにリモート・メソッドのスタブ・クラス・インスタンスを得てロードする場合、正しいスタブ・クラス・インスタンスが見つかるまで呼出しが遅延する。メソッドの参照を受信した場合でもメソッドが実際には呼び出されない場合、スタブ・クラス・インスタンスを見つけてロードする必要はない。

【００３６】当然のことながら、本発明によるシステムが、任意の部分を適当なプログラムによって制御することができる、特殊目的ハードウェアまたは汎用コンピュータ・システム、あるいはそれらの組合せで全体的または部分的に構築することができる。プログラムは、従来どおり、システムの一部を全体的または部分的に構成することも、あるいは全体的または部分的にシステム上に記憶することもでき、あるいは従来どおりに情報を転送するネットワークまたはその他の機構上のシステムに全体的または部分的に設けることができる。また、当然のことながら、システムは、システムに直接接続することも、あるいは従来どおりに情報を転送するネットワークまたはその他の機構上のシステムへ情報を転送することもできる、オペレータがオペレータ入力要素（図示せず）を使って与えた情報によって操作し、あるいはその他の方法で制御することができる。

【００３７】前述の説明は、本発明の特定の実施形態に制限されている。しかし、本発明に様々な変形および修正を加えて本発明のいくつかまたはすべての利点を達成することができることは明らかである。添付の特許請求の範囲の目的は、このような変形および修正、ならびに本発明の真の趣旨および範囲内の他のそのような変形および修正をカバーすることである。

【図面の簡単な説明】

【図１】「スタブ」情報の動的ローディングおよび使用を容易にし、実行して、あるアドレス空間で動作するプログラムが他のアドレス空間内のリモート・メソッドまたは手続きの処理を呼び出すことができるようにするために本発明によって構築された構成を含むコンピュータ・ネットワークの機能ブロック図である。

【図２】本発明を理解する際に有用な、図１に示した構成によって実行される動作を示すフローチャートであり、スタブ情報の動的ローディングを得ることに関連して実行される動作を示す図である。

【図３】本発明を理解する際に有用な、図１に示した構成によって実行される動作を示すフローチャートであり、スタブ情報の動的ローディングを得ることに関連して実行される動作を示す図である。

【図４】本発明を理解する際に有用な、図１に示した構成によって実行される動作を示すフローチャートであり、スタブ情報の動的ローディングを得ることに関連して実行される動作を示す図である。

【図５】本発明を理解する際に有用な、図１に示した構成によって実行される動作を示すフローチャートであり、スタブ情報を使用してリモート・メソッドまたは手続きの処理を呼び出すことに関連して実行される動作を示す図である。

【図６】本発明を理解する際に有用な、図１に示した構成によって実行される動作を示すフローチャートであり、スタブ情報を使用してリモート・メソッドまたは手続きの処理を呼び出すことに関連して実行される動作を示す図である。

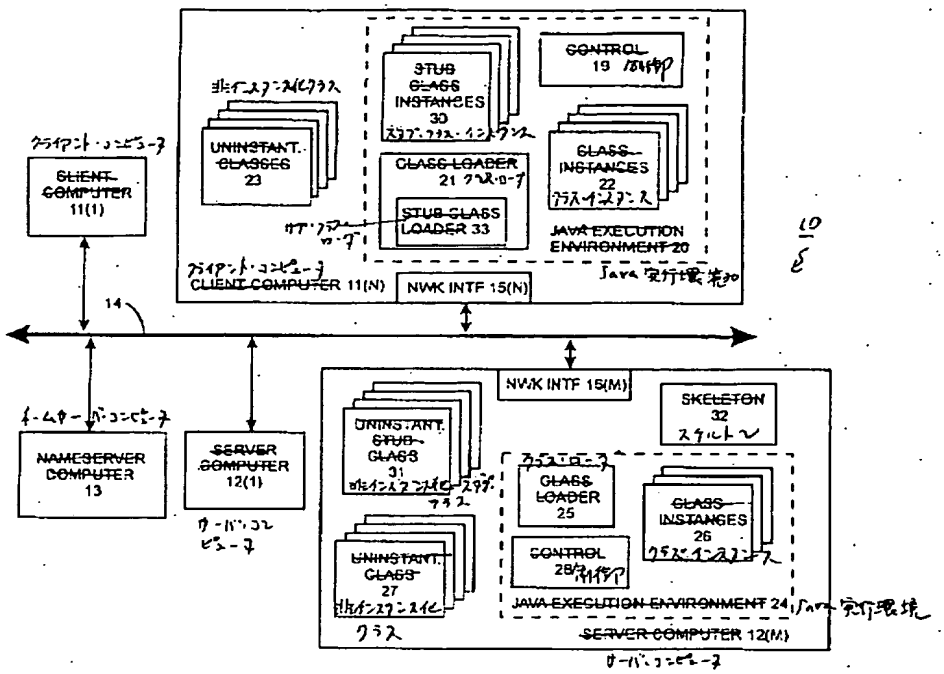
【符号の説明】

１０　コンピュータ・ネットワーク
１１　クライアント・コンピュータ
１２　サーバ・コンピュータ
１３　ネームサーバ・コンピュータ
１４　通信リンク
１５、１６　ネットワーク・インタフェース装置
２０、２４　実行環境
２１、２５、３３　クラス・ローダ
２２、２６　クラス・インスタンス
２３、２７　非インスタンス化クラス
１９、２８　制御モジュール
３０、３１　スタブ・クラス

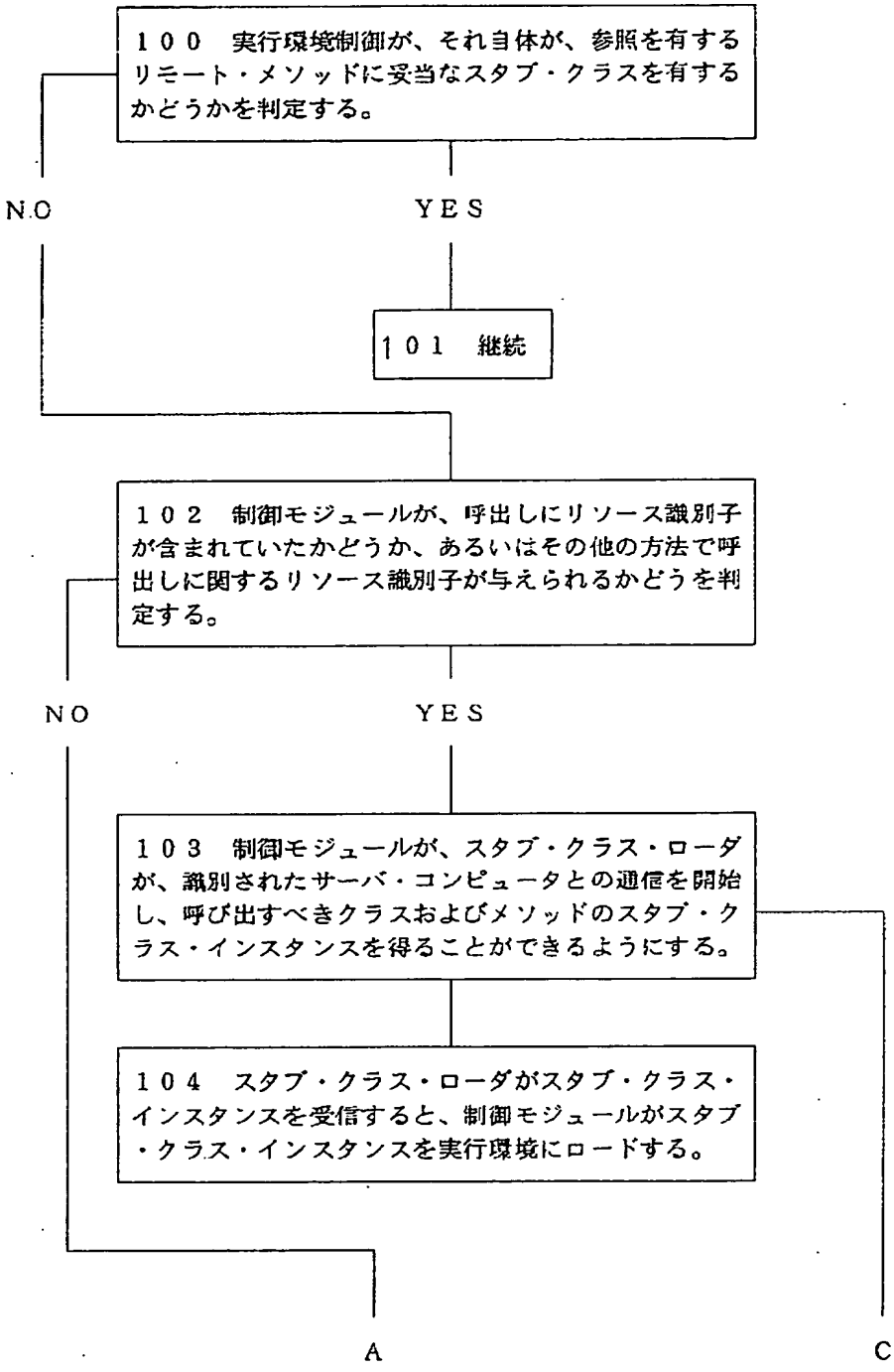
21

３２　スケルトン

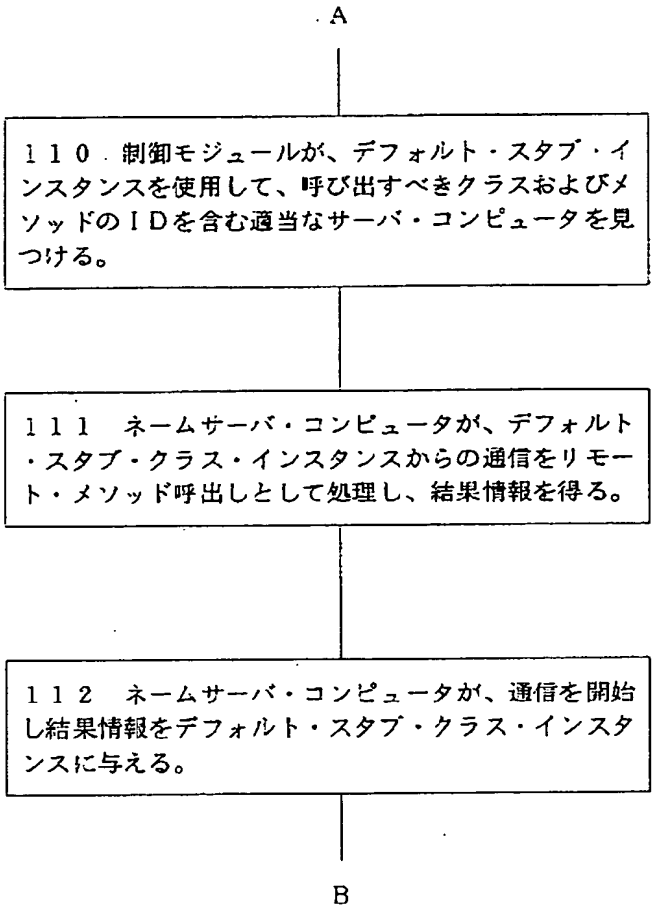
【図１】



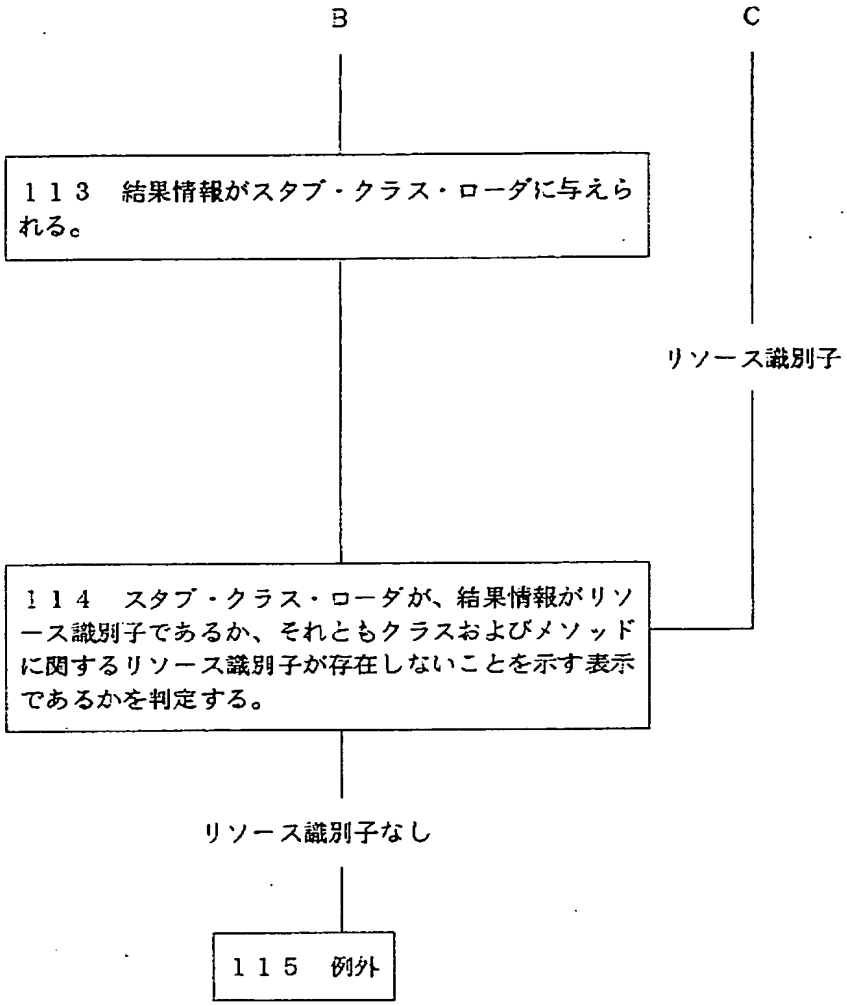
【図2】


100　実行環境制御が、それ自体が、参照を有するリモート・メソッドに妥当なスタブ・クラスを有するかどうかを判定する。
NO
YES
101　継続
102　制御モジュールが、呼出しにリソース識別子が含まれていたかどうか、あるいはその他の方法で呼出しに関するリソース識別子が与えられるかどうを判定する。
NO
YES
103　制御モジュールが、スタブ・クラス・ローダが、識別されたサーバ・コンピュータとの通信を開始し、呼び出すべきクラスおよびメソッドのスタブ・クラス・インスタンスを得ることができるようにする。
104　スタブ・クラス・ローダがスタブ・クラス・インスタンスを受信すると、制御モジュールがスタブ・クラス・インスタンスを実行環境にロードする。
A
C

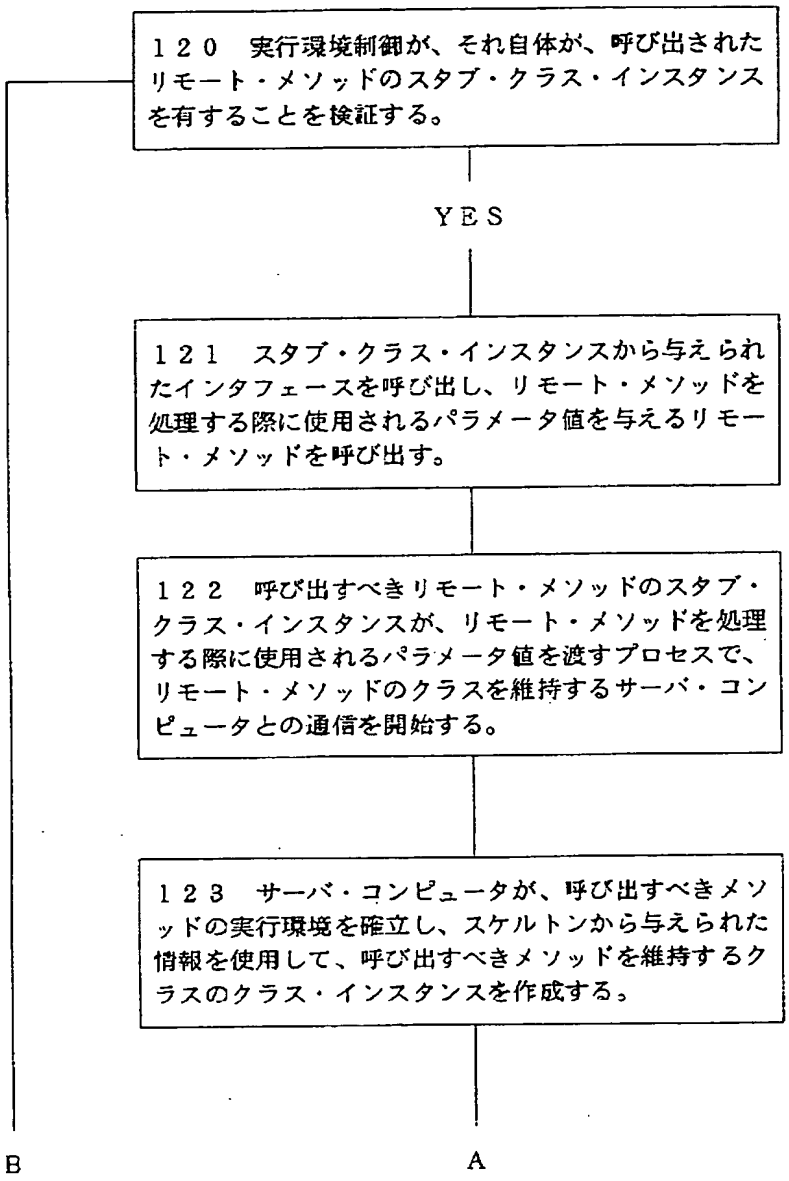
【図３】


A
１１０　制御モジュールが、デフォルト・スタブ・インスタンスを使用して、呼び出すべきクラスおよびメソッドのＩＤを含む適当なサーバ・コンピュータを見つける。
１１１　ネームサーバ・コンピュータが、デフォルト・スタブ・クラス・インスタンスからの通信をリモート・メソッド呼出しとして処理し、結果情報を得る。
１１２　ネームサーバ・コンピュータが、通信を開始し結果情報をデフォルト・スタブ・クラス・インスタンスに与える。
B

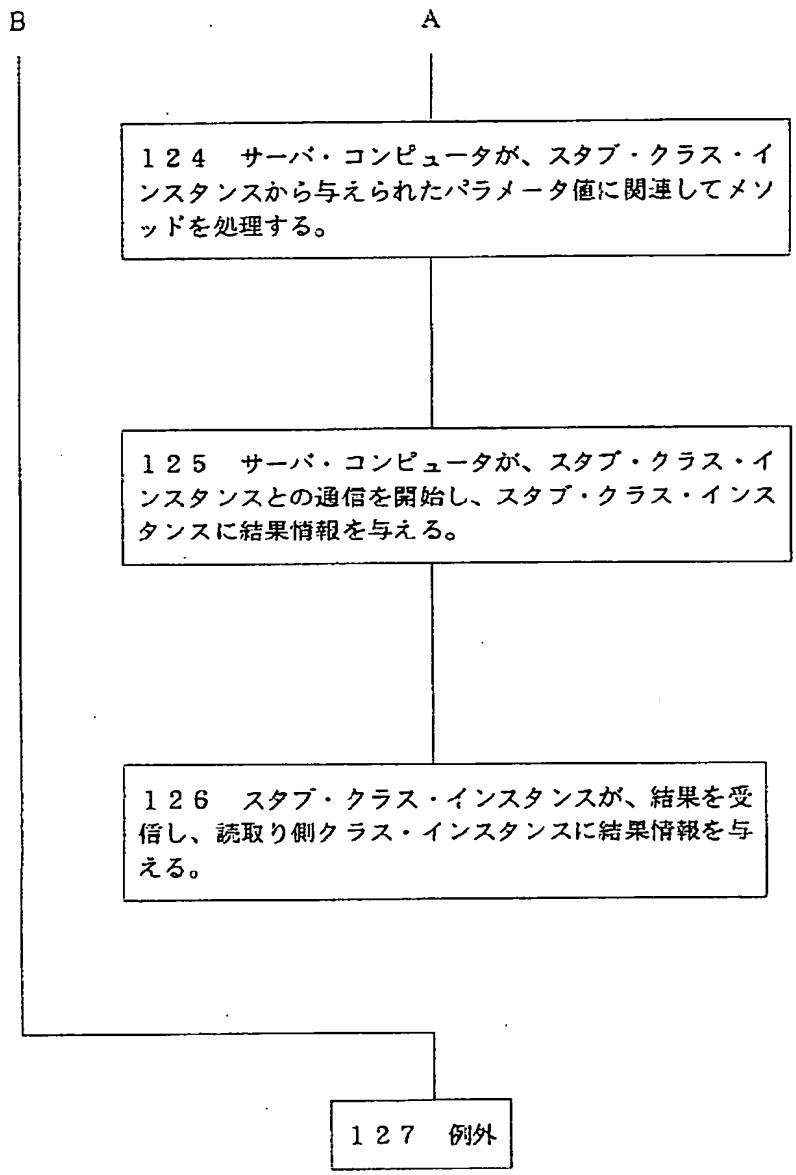
【図４】


B
C
１１３　結果情報がスタブ・クラス・ローダに与えられる。
リソース識別子
１１４　スタブ・クラス・ローダが、結果情報がリソース識別子であるか、それともクラスおよびメソッドに関するリソース識別子が存在しないことを示す表示であるかを判定する。
リソース識別子なし
１１５　例外

【図５】

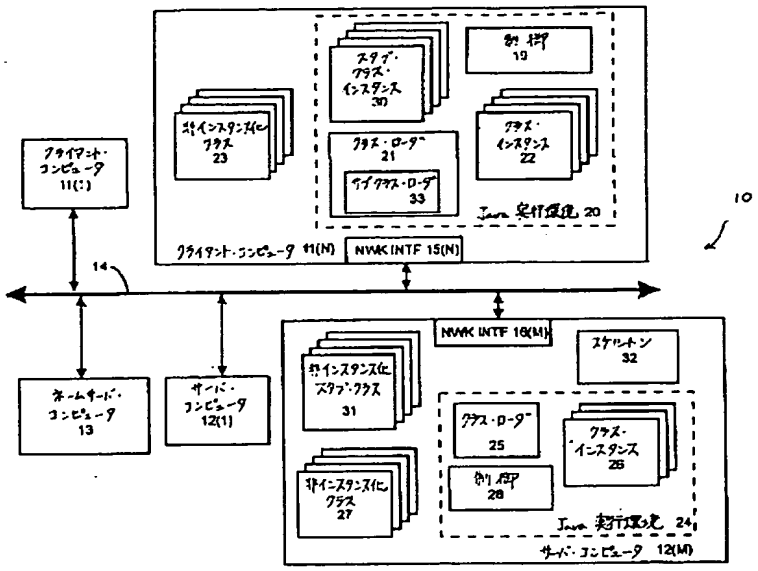

１２０　実行環境制御が、それ自体が、呼び出されたリモート・メソッドのスタブ・クラス・インスタンスを有することを検証する。
YES
１２１　スタブ・クラス・インスタンスから与えられたインタフェースを呼び出し、リモート・メソッドを処理する際に使用されるパラメータ値を与えるリモート・メソッドを呼び出す。
１２２　呼び出すべきリモート・メソッドのスタブ・クラス・インスタンスが、リモート・メソッドを処理する際に使用されるパラメータ値を渡すプロセスで、リモート・メソッドのクラスを維持するサーバ・コンピュータとの通信を開始する。
１２３　サーバ・コンピュータが、呼び出すべきメソッドの実行環境を確立し、スケルトンから与えられた情報を使用して、呼び出すべきメソッドを維持するクラスのクラス・インスタンスを作成する。
B
A

【図６】

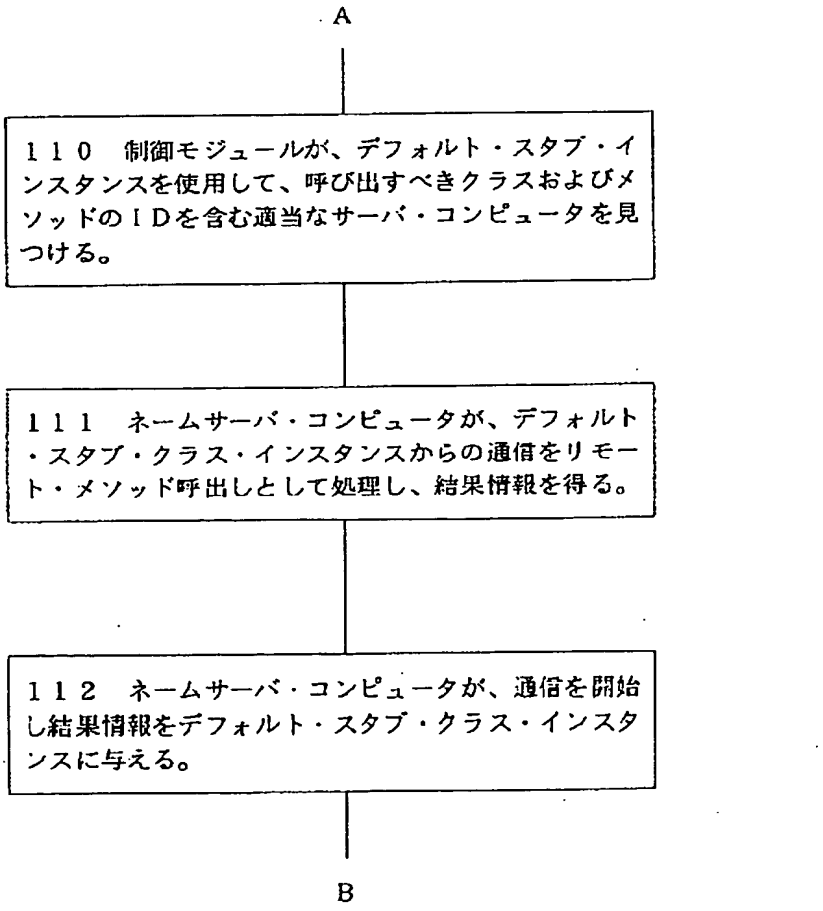


【手続補正書】
【提出日】平成９年７月１５日
【手続補正１】
【補正対象書類名】図面
【補正対象項目名】全図
【補正方法】変更
【補正内容】

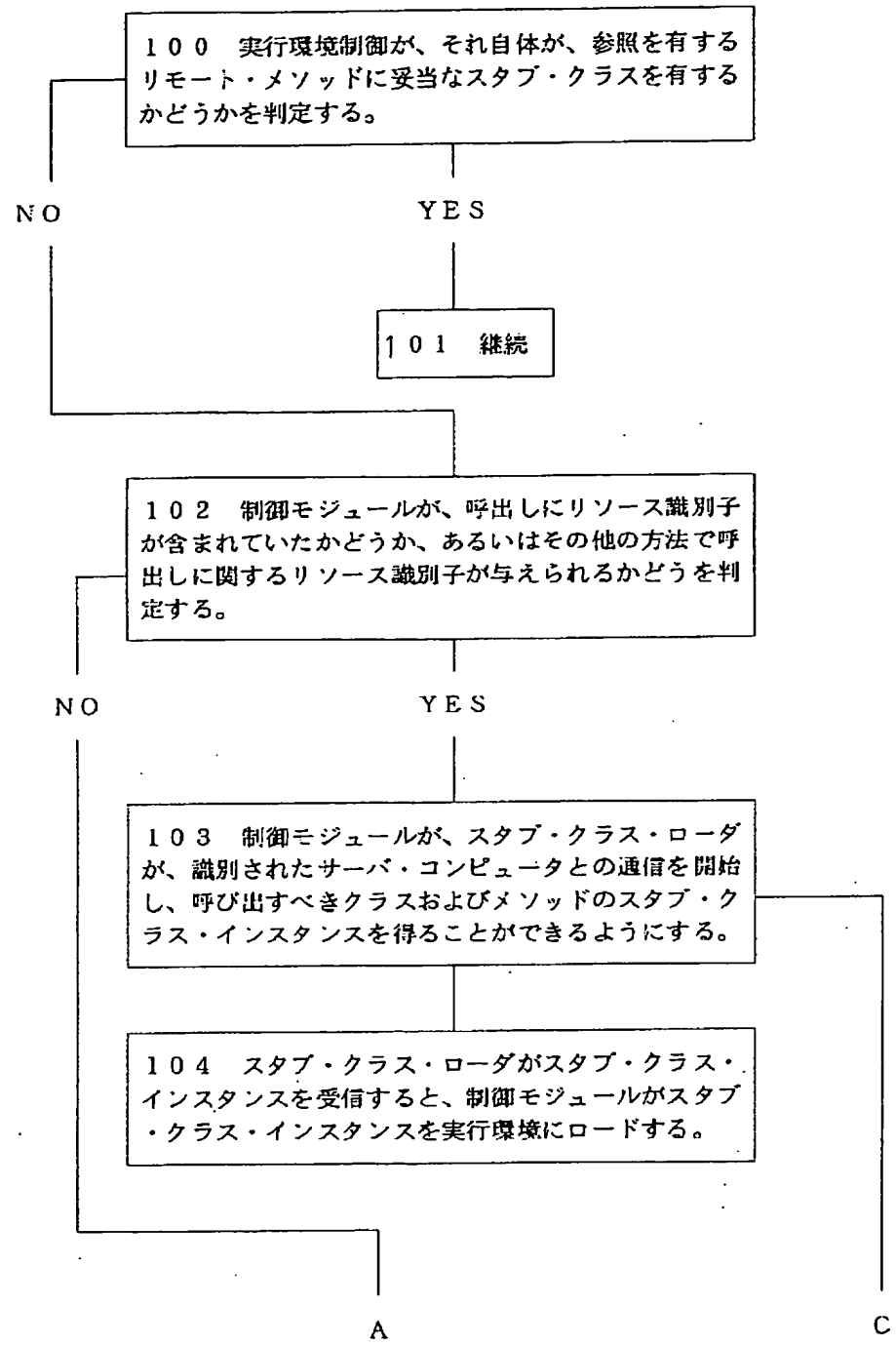
【図　１】


スタブ・クラス・インスタンス 30
制御部 19
非インスタンス化クラス 23
クラス・ローダ 21
クラス・インスタンス 22
サイクラス・ローダ 33
Java 実行環境 20
クライアント・コンピュータ 11(N)
NWK INTF 15(N)
10
14
クライアント・コンピュータ 11(1)
NWK INTF 16(M)
スケルトン 32
非インスタンス化スタブ・クラス 31
クラス・ローダ 25
クラス・インスタンス 26
制御部 28
非インスタンス化クラス 27
Java 実行環境 24
ネームサーバ・コンピュータ 13
サーバ・コンピュータ 12(1)
サーバ・コンピュータ 12(M)

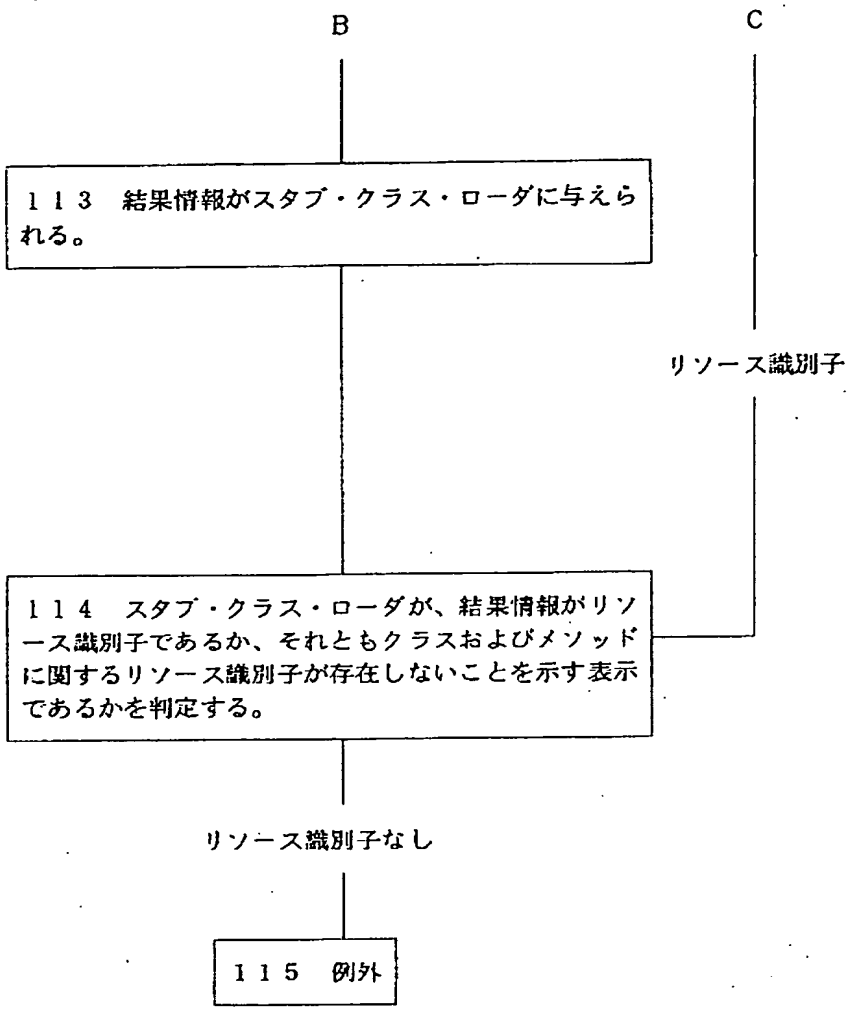
【図　３】


A
１１０　制御モジュールが、デフォルト・スタブ・インスタンスを使用して、呼び出すべきクラスおよびメソッドのＩＤを含む適当なサーバ・コンピュータを見つける。
１１１　ネームサーバ・コンピュータが、デフォルト・スタブ・クラス・インスタンスからの通信をリモート・メソッド呼出しとして処理し、結果情報を得る。
１１２　ネームサーバ・コンピュータが、通信を開始し結果情報をデフォルト・スタブ・クラス・インスタンスに与える。
B

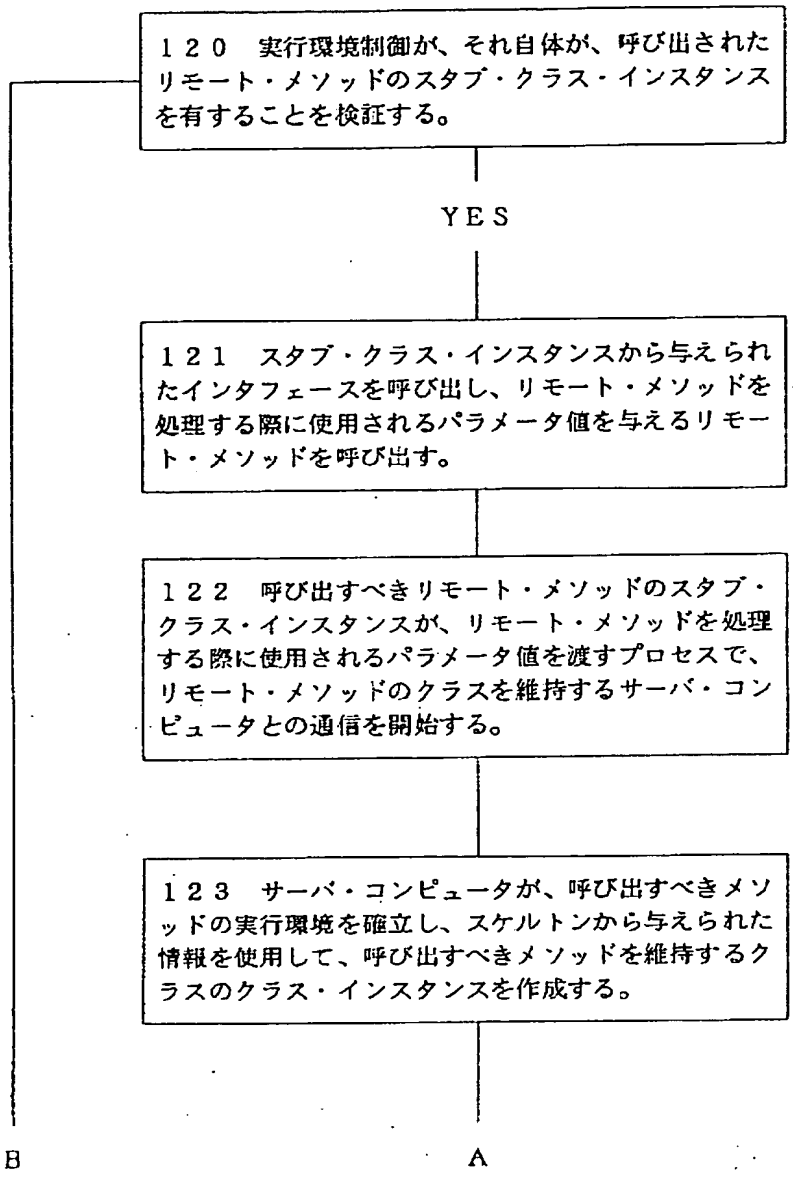
【図　２】


１００　実行環境制御が、それ自体が、参照を有するリモート・メソッドに妥当なスタブ・クラスを有するかどうかを判定する。
NO
YES
１０１　継続
１０２　制御モジュールが、呼出しにリソース識別子が含まれていたかどうか、あるいはその他の方法で呼出しに関するリソース識別子が与えられるかどうを判定する。
NO
YES
１０３　制御モジュールが、スタブ・クラス・ローダが、識別されたサーバ・コンピュータとの通信を開始し、呼び出すべきクラスおよびメソッドのスタブ・クラス・インスタンスを得ることができるようにする。
１０４　スタブ・クラス・ローダがスタブ・クラス・インスタンスを受信すると、制御モジュールがスタブ・クラス・インスタンスを実行環境にロードする。
A
C

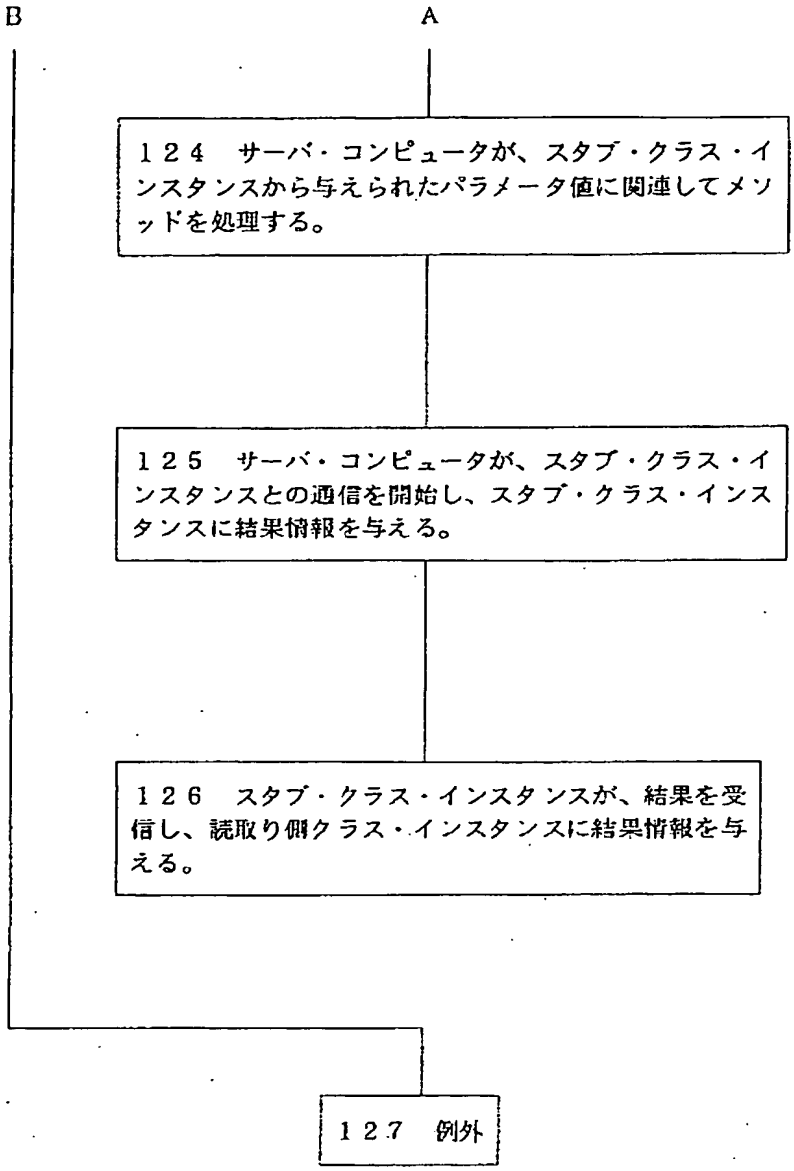
【図　４】


B
C
１１３　結果情報がスタブ・クラス・ローダに与えられる。
リソース識別子
１１４　スタブ・クラス・ローダが、結果情報がリソース識別子であるか、それともクラスおよびメソッドに関するリソース識別子が存在しないことを示す表示であるかを判定する。
リソース識別子なし
１１５　例外

【図　５】


１２０　実行環境制御が、それ自体が、呼び出されたリモート・メソッドのスタブ・クラス・インスタンスを有することを検証する。
YES
１２１　スタブ・クラス・インスタンスから与えられたインタフェースを呼び出し、リモート・メソッドを処理する際に使用されるパラメータ値を与えるリモート・メソッドを呼び出す。
１２２　呼び出すべきリモート・メソッドのスタブ・クラス・インスタンスが、リモート・メソッドを処理する際に使用されるパラメータ値を渡すプロセスで、リモート・メソッドのクラスを維持するサーバ・コンピュータとの通信を開始する。
１２３　サーバ・コンピュータが、呼び出すべきメソッドの実行環境を確立し、スケルトンから与えられた情報を使用して、呼び出すべきメソッドを維持するクラスのクラス・インスタンスを作成する。
B
A

【図　６】



フロントページの続き

(71)出願人　591064003
901　SAN　ANTONIO　ROAD
PALO　ALTO, CA　94303, U. S. A.

(72)発明者　ジェームズ・エイチ・ワルド
アメリカ合衆国・01826・マサチューセッツ州・ドラカット・ルビー　ロード・155

(72)発明者　ロジャー・リッグス
アメリカ合衆国・01803・マサチューセッツ州・バーリントン・ブライアウッド　レーン・4